AD-A049 062 NAVAL RESEARCH LAB WASHINGTON D C F/G 6/4
EXPERIMENTS WITH SOME ALGORITHMS THAT FIND CENTRAL SOLUTIONS FO--ETC(U)
SEP 77 J R SLAGLE

UNCLASSIFIED NRL-MR-3609 SBIE-AD-E000 050 NL

| OF |
AD
A049062

END
DATE
FILMED
2-78
DDC

NRL Memorandum Report 3609

Experiments with Some Algorithms that Find Central Solutions for Pattern Classification

JAMES R. SLAGLE

Computer Science Laboratory
Communications Science Division

September 1977

NAVAL RESEARCH LABORATORY
Washington, D.C.

(9) Interin rept.,

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: NRL Memorandum Report 3609	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): (6) EXPERIMENTS WITH SOME ALGORITHMS THAT FIND CENTRAL SOLUTIONS FOR PATTERN CLASSIFICATION.	5. TYPE OF REPORT & PERIOD COVERED: Interim report on a continuing NRL problem.
	6. PERFORMING ORG. REPORT NUMBER: (14) NRL-MR-3609
7. AUTHOR(s): (10) James R. Slagle	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Research Laboratory ✓ Washington, D.C. 20375	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NRL Problem B02-23 61153N-14, RR014-02, RR014-02-41
11. CONTROLLING OFFICE NAME AND ADDRESS: (16) RR014 02	12. REPORT DATE: (11) Sept~~ember~~ 1977
	13. NUMBER OF PAGES: ~~23~~ (12) 24p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): (12) RR014 0241 (13) SBIE	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report): Approved for public release; distribution unlimited. (19) AD-E000 050	DDC RECEIVED JAN 30 1978 B
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	
18. SUPPLEMENTARY NOTES	

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Pattern recognition	Centering	Linearly separable
Pattern classification	Centrality criteria	Relaxation algorithm
Linear discriminants	Dead zone	Accelerated relaxation
Central hyperplanes	Hyperplane	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In two-class pattern recognition, it is a standard technique to have an algorithm for finding hyperplanes which separates the two classes in a linearly separable training set. The traditional methods find a hyperplane which separates all points in one class from all points in the other, but such a hyperplane is not necessarily centered in the

(Continues)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

251 950 B

20. Abstract (Continued)

empty space between the two classes. Since a central hyperplane does not favor one class or the other, it should have a lower error rate in classifying new points and is therefore better than a non-central hyperplane. Six algorithms for finding central hyperplanes are tested on three data sets. Although frequently used in practice, the modified relaxation algorithm is very poor. Three algorithms, which are defined in the paper, are found to be quite good.

CONTENTS

EXPERIMENTS WITH SOME ALGORITHMS THAT FIND CENTRAL SOLUTIONS FOR PATTERN CLASSIFICATION

1. INTRODUCTION

Linear discriminant functions (hyperplanes) play a very important part in automatic pattern recognition [1,2,3,4,5,7,8]. This paper explores a number of improved methods of finding hyperplanes which are better because they are more nearly central between two different classes. This is sometimes called the problem of finding central hyperplanes.

To consider this problem in more detail, suppose we are given a linearly separable training set of pattern vectors in two classes. "Linearly separable" means that the two classes can be separated by some hyperplane. There are many algorithms that will find a solution hyperplane, however such methods guarantee only that all points of one class from the training set will be on one side of the hyperplane and that all points of the other class will be on the other side. But such a hyperplane is not necessarily centrally located in the empty space between the two classes.

In order to understand why a central hyperplane is desirable we must remember that the given points (the training set) are only a sample of the universe of points that we want to classify. The purpose of the hyperplane is to classify new points (test vectors) from the universe. When a new point is given, common sense tells us to assign it to that class which its nearby neighbors belong to. Only a central hyperplane will do this.

Figures 1 and 2 show an example in two dimensions. The A and B points are the training set. Figure 1 shows a poor solution hyperplane and Figure 2 shows a much better (more central) solution hyperplane. X represents a new point. Common sense tells us that X should be in class B, but the poor hyperplane puts it into class A. The good hyperplane of Figure 2 correctly puts X into class B.

Of course, the ideal way to solve this kind of problem is to fit a probability density function to each class by some process of statistical inference, and then choose the surface of equal probability density as the decision boundry between the two classes. However, such a surface is not necessarily flat and the process of finding such a surface may be complex and expensive.

Note: Manuscript submitted September 7, 1977.

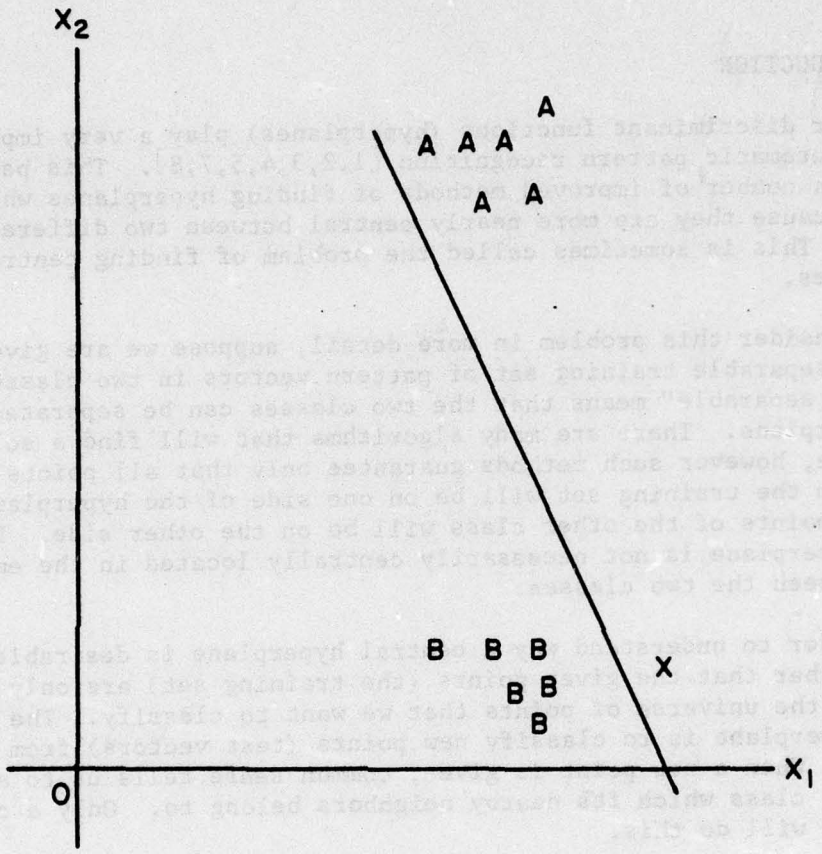

Fig. 1 - A poor solution line for a training set of two-dimensional pattern vectors

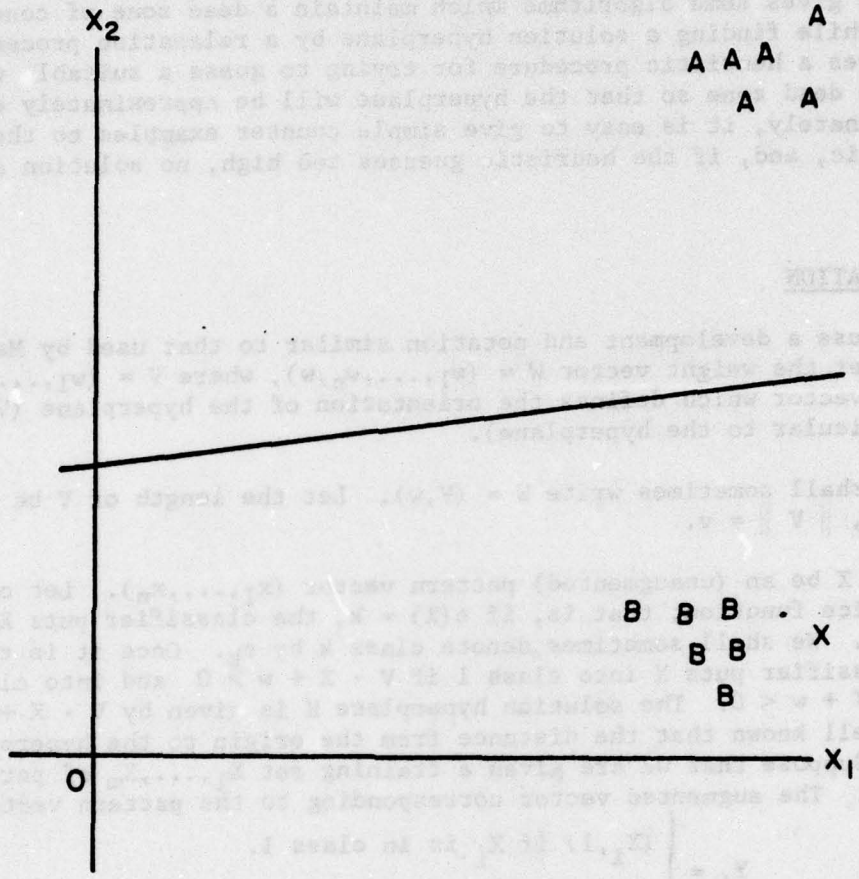

Fig. 2 - A central solution line for the same training set as in Fig. 1

Therefore finding a central hyperplane is a simple approximation which is adequate for many practical applications. In fact the central hyperplane may often be the optimal solution to a practical pattern recognition problem, when economic factors are considered.

Slagle [7] discusses the problem of finding a central hyperplane and presents an algorithm which is the forerunner of algorithm E below. Yue [8] gives some algorithms which maintain a dead zone of constant width while finding a solution hyperplane by a relaxation process. Yue gives a heuristic procedure for trying to guess a suitable width for the dead zone so that the hyperplane will be approximately centered. Unfortunately, it is easy to give simple counter examples to the heuristic, and, if the heuristic guesses too high, no solution exists at all.

2. NOTATION

We use a development and notation similar to that used by Meisel [5]. Let the weight vector $W = (w_1,\ldots,w_n,w)$, where $V = (w_1,\ldots,w_n)$ is the vector which defines the orientation of the hyperplane (V is perpendicular to the hyperplane).

We shall sometimes write $W = (V,w)$. Let the length of V be v, that is, $\| V \| = v$.

Let X be an (unaugmented) pattern vector $(x_1,\ldots,x_n)$. Let c(X) be the choice function; that is, if $c(X) = k$, the classifier puts X into class k. We shall sometimes denote class k by c_k. Once it is trained, our classifier puts X into class 1 if $V \cdot X + w > 0$ and into class 2 if $V \cdot X + w < 0$. The solution hyperplane H is given by $V \cdot X + w = 0$. It is well known that the distance from the origin to the hyperplane is $-w/v$. Suppose that we are given a training set $X_1,\ldots,X_m$ of pattern vectors. The augmented vector corresponding to the pattern vector X_i is

$$Y_i = \begin{cases} (X_i,1) \text{ if } X_i \text{ is in class 1.} \\ (-X_i,-1) \text{ if } X_i \text{ is in class 2.} \end{cases}$$

Now let us consider the dead zone. Roughly speaking, this is the zone surrounding the hyperplane which contains no points from the training set.

Let b be a positive real number. We chose $b = 1$ in our experiments. b is related to the dead zone width as follows. We want to train our classifier (that is, find a W) so that it puts X_i into class 1 if $V \cdot X_i + w \geq b$ and into class 2 if $V \cdot X_i + w \leq -b$. See Fig. 3.

This can be accomplished by training the classifier so that $W \cdot Y_i \geq b$ for all $i = 1,\ldots,m$. That is, we want $e_i \geq 0$ where

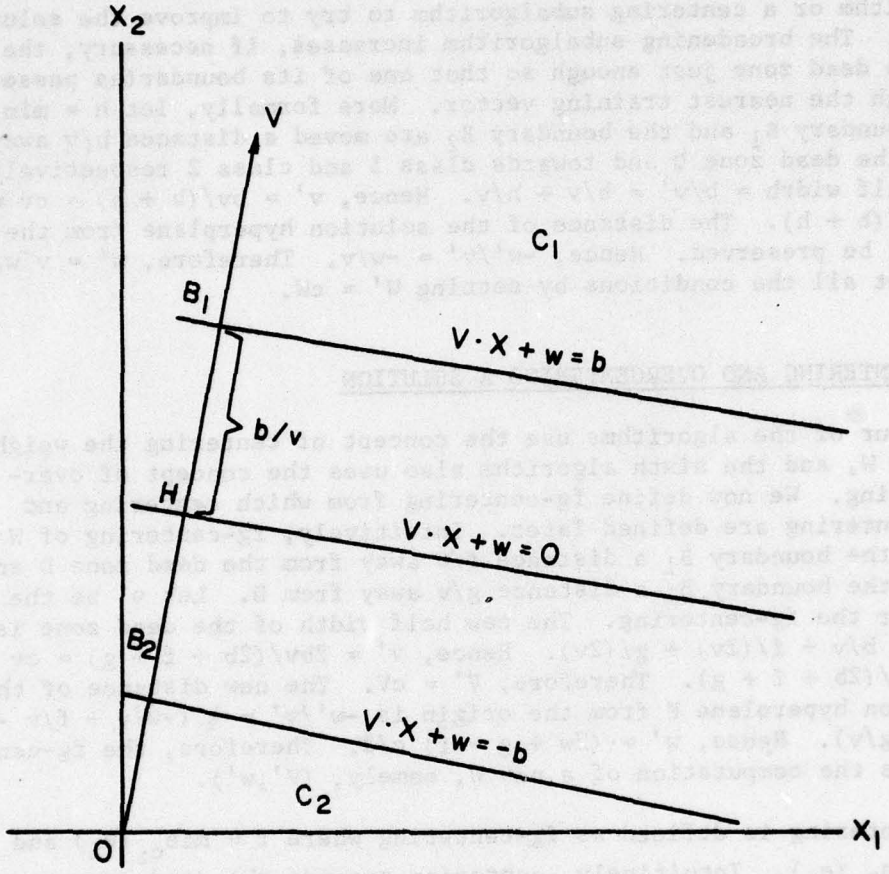

Fig. 3 - A solution hyperplane H and dead zone in two dimensions

$e_i = W \cdot Y_i - b$. Recall that e_i/v is the n-dimensional distance from the pattern X_i to the appropriate hyperplane dead zone boundary. Half of the dead zone width is b/v.

3. BROADENING THE DEAD ZONE OF A SOLUTION HYPERPLANE

Each of the six algorithms we shall describe uses a broadening subalgorithm or a centering subalgorithm to try to improve the solution it finds. The broadening subalgorithm increases, if necessary, the breadth of the dead zone just enough so that one of its boundaries passes through the nearest training vector. More formally, let $h = \min_i e_i$. The boundary B_1 and the boundary B_2 are moved a distance h/v away from the dead zone D and towards class 1 and class 2 respectively. The new half width = $b/v' = b/v + h/v$. Hence, $v' = bv/(b + h) = cv$ where $c = b/(b + h)$. The distance of the solution hyperplane from the origin should be preserved. Hence, $-w'/v' = -w/v$. Therefore, $w' = v'w/v = cw$. We meet all the conditions by setting $W' = cW$.

4. CENTERING AND OVERCENTERING A SOLUTION

Four of the algorithms use the concept of centering the weight vector W, and the sixth algorithm also uses the concept of overcentering. We now define fg-centering from which centering and overcentering are defined later. Intuitively, fg-centering of W moves the boundary B_1 a distance f/v away from the dead zone D and moves the boundary B_2 a distance g/v away from D. Let v' be the new v after the fg-centering. The new half width of the dead zone is $b/v' = b/v + f/(2v) + g/(2v)$. Hence, $v' = 2bv/(2b + f + g) = cv$ where $c = 2b/(2b + f + g)$. Therefore, $V' = cV$. The new distance of the solution hyperplane H from the origin is $-w'/v' = \frac{1}{2}(-w/v + f/v - w/v - g/v)$. Hence, $w' = (2w + g - f)\, c/2$. Therefore, the fg-centering of W is the computation of a new W, namely, (V',w').

Centering is defined as fg-centering where $f = \min_{c_1}(e_i)$ and $g = \min_{c_2}(e_i)$. Intuitively, centering expands the dead zone as much as possible until it bumps into a training point (vector) in c_1 and a training point in c_2.

5. THE ALGORITHMS TO BE TESTED

The six algorithms to be compared are broadened modified relaxation, centered modified relaxation, broadened accelerated relaxation, centered accelerated relaxation, extended centered accelerated relaxation, and centered overcentered accelerated relaxation. The first two are based on modified relaxation [2,4,5]. The next two are based on accelerated relaxation [1]. The last two we introduce here.

A. Broadened Modified Relaxation. This algorithm examines the training set pattern vectors Y_i one at a time. Let W_k be the weight vector after the k^{th} examination. Let W_0 be arbitrary, for example, $W_0 = (0,0,\ldots,0)$.

$$W_{k+1} = \begin{cases} W_k - p\ (e_i - c)y_i/s_i & \text{if } e_i < 0 \\ W_k & \text{otherwise} \end{cases}$$

where $c > 0$ and $s_i = \|y_i\|^2$.

The algorithm continues until the training set is linearly separated or has been examined P times. When a solution is found, it is broadened. This algorithm (without the broadening) is often used in practice, because it can be proved [2,4,5] that, if the two classes in the training set are linearly separable, the algorithm will find a solution in a finite number of steps. In our implementation, we set $P = 100$, $c = 0.1$, and, with the help of advice from [4], $p = 1.999$. If c is dropped from the above equation, this algorithm without the broadening becomes the (unmodified) relaxation algorithm.

B. Centered Modified Relaxation. This is the same as broadened modified relaxation, except that the solution is centered at the end instead of broadened.

C. Broadened Accelerated Relaxation. The accelerated relaxation algorithm due to C. L. Chang [1] has two parts, the relaxation phase and the acceleration phase. These two parts are repeated alternately. The process begins with an arbitrary initial weight vector W_0^A. After one complete pass through the set of all training vectors, W_0^A becomes W_1^R, in general W_ℓ^A becomes $W_{\ell+1}^R$. The relaxation process of algorithm A is used during this pass. W^R is the result of the relaxation process and W^A is the result of the acceleration process.

During the acceleration phase $W_{\ell+1}^R$ becomes $W_{\ell+1}^A$. This is done by constructing a line from the point W_ℓ^A to $W_{\ell+1}^R$ and then extending this line beyond $W_{\ell+1}^R$. A search is then made along this extension for the interval of best separation score. $W_{\ell+1}^A$ is the center of the interval. The procedure is repeated up to 30 times or until it converges (which means that the training set is linearly separated). In general, accelerated relaxation is much faster than modified relaxation [1]. Our algorithm broadens the solution found by the accelerated relaxation algorithm.

D. Centered Accelerated Relaxation. This is the same as the broadened accelerated relaxation algorithm, except that the solution is centered at the end instead of broadened.

E. Extended Centered Accelerated Relaxation. The resulting weight vector from centered accelerated relaxation (algorithm D) is given as the initial weight vector to the modified extended relaxation algorithm given by equation (3) below with c = 0.1, p = 1.999, and the number P of passes through the training set being ten. The weight vector obtained is then centered. Thus, the major part of this algorithm is the modified extended relaxation algorithm, which is described in the rest of this section.

Before describing the modified extended relaxation algorithm precisely, we discuss the intuitive idea upon which it is based. The relaxation algorithm (See Algorithm A Above) is frequently used in practice. It obtains a discriminating hyperplane by trying to minimize the (approximate) relaxation risk R where R is the relaxation loss averaged over the training set. We now present a very simple pattern recognition training set in order to illustrate the advantage of our approach (extended relaxation) over the relaxation approach. Consider the two-class two-pattern one-dimensional pattern recognition training set shown in Fig. 4. It consists of two patterns (points) $X_1 = 2$ and $X_2 = 6$ with X_1 in class 1 and X_2 in class 2. Intuitively, the best point to divide class 1 from class 2 is $X = 4$, the midpoint of the interval from X_1 to X_2. The classifier should classify a pattern from the test set into class 1 if it lies to the left of $X = 4$ and into class 2 if to the right. It turns out that our approach yields $X = 4$, whereas the relaxation approach may choose the dividing point X anywhere in the open interval (2,6). Our particular implementation of relaxation (algorithm A) chooses $X = 3.0$, which is poor.

For the simple training set we are considering, Fig. 5 shows the part of the relaxation loss (for the training set) due to the position of the left boundary B_1 of the dead zone. We are considering the simple case when the loss is taken to be proportional to the square of the distance between B_1 and X_1, when X_1 is on the wrong side of B_1; the loss is zero, when X_1 is on the correct side of B_1. Fig. 6 shows the part of the loss due to the position of the right boundary B_2 of the dead zone. It is clear that the total loss is zero as long as both dead zone boundaries are in the closed interval $[2,6]$. Therefore, the dividing point X may be any point in the open interval (2,6).

In our new (extended relaxation) approach, we use a loss function like the one shown in Fig. 7 and Fig. 8. Instead of being zero when X_1 is on the correct side of B_1, the loss is a small positive fraction of the square of the distance between B_1 and X_1. This and the similar fact about X_2 and B_2 account for the two gently sloping semiparabolas in Fig. 7 and Fig. 8 respectively. It is clear that the losses are minimized (zero in fact) only when the dead zone boundaries are $B_1 = 2$ and $B_2 = 6$. This yields the central dividing point $X = 4$ and its advantages. We next generalize these ideas to more points and higher dimensions.

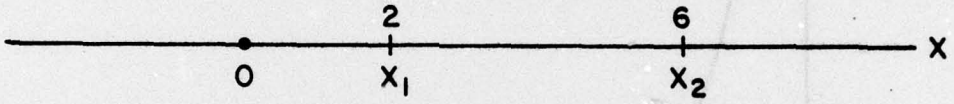

Fig. 4 - A two-class two-pattern one-dimensional pattern recognition training set

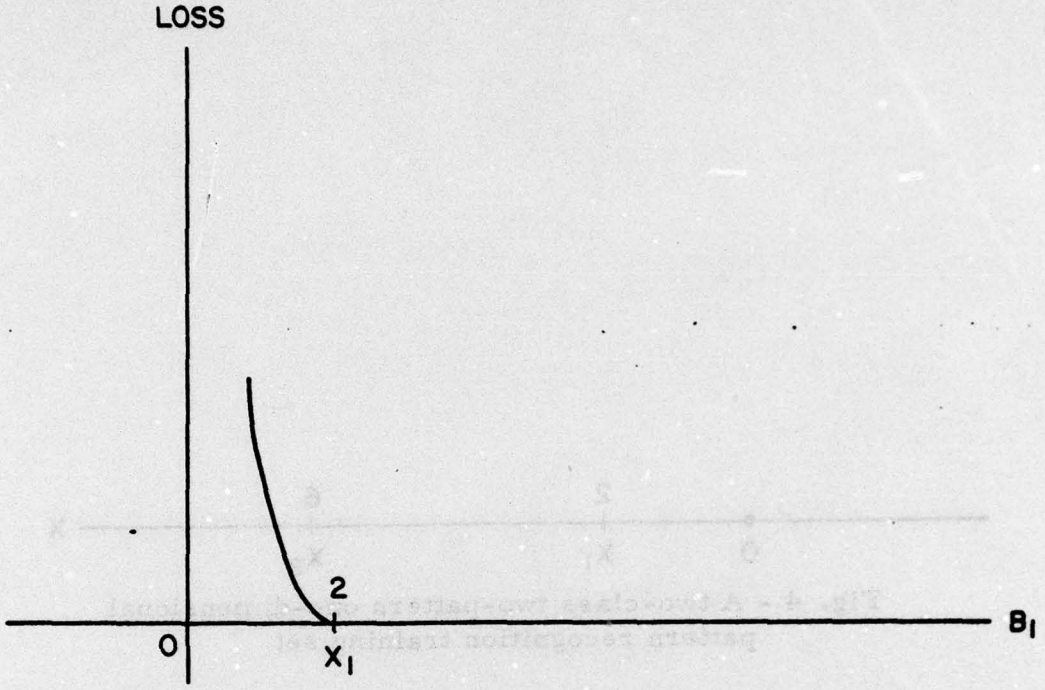

Fig. 5 - Portion of the relaxation loss (for the training set) as a function of the position of the left boundary B_1 of the dead zone

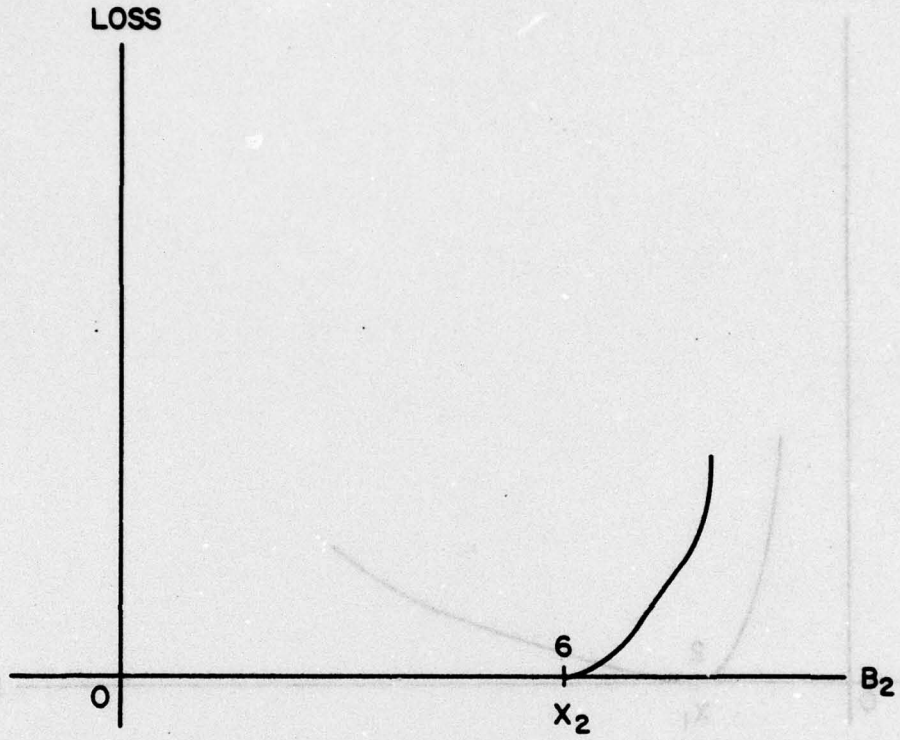

Fig. 6 - Portion of the relaxation loss as a function of the position of the right boundary B_2 of the dead zone

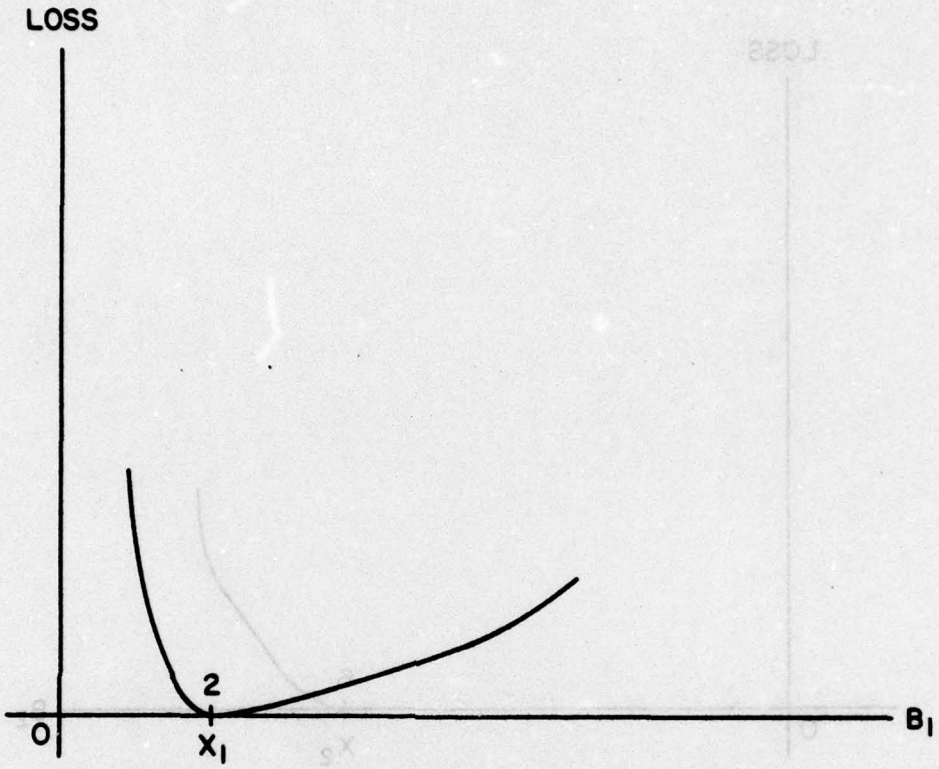

Fig. 7 - Portion of the extended relaxation loss as a function of the position of the left boundary B_1 of the dead zone

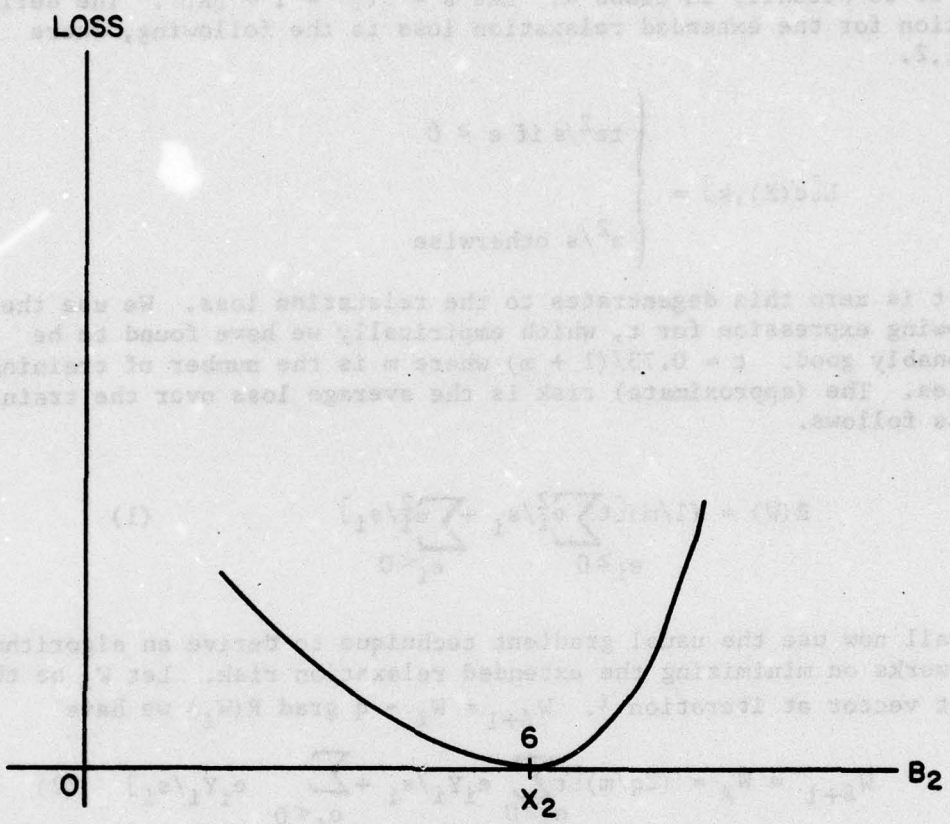

Fig. 8 - Portion of the extended relaxation loss as a function of the position of the right boundary B_2 of the dead zone

We now define the extended relaxation loss and derive the (approximate) extended relaxation risk from this loss. We shall then derive the extended relaxation algorithm as an algorithm that works on minimizing this risk. Finally, we modify the extended relaxation algorithm to obtain the modified extended relaxation algorithm. Let $L[j,k]$ be the loss associated with classifying a pattern into class j when it is actually in class k. Let $s = \|Y\|^2 = 1 + \|X\|^2$. The defining equation for the extended relaxation loss is the following, where $k = 1,2$.

$$L[c(X),k] = \begin{cases} te^2/s \text{ if } e \geq 0 \\ e^2/s \text{ otherwise} \end{cases}$$

When t is zero this degenerates to the relaxation loss. We use the following expression for t, which empirically we have found to be reasonably good. $t = 0.73/(1 + m)$ where m is the number of training samples. The (approximate) risk is the average loss over the training set as follows.

$$R(W) = (1/m)[t\sum_{e_i \geq 0} e_i^2/s_i + \sum_{e_i < 0} e_i^2/s_i] \qquad (1)$$

We shall now use the usual gradient technique to derive an algorithm that works on minimizing the extended relaxation risk. Let W_ℓ be the weight vector at iteration ℓ. $W_{\ell+1} = W_\ell - q \text{ grad } R(W_\ell)$ we have

$$W_{\ell+1} = W_\ell = (2q/m)[t\sum_{e_i \geq 0} e_i Y_i/s_i + \sum_{e_i < 0} e_i Y_i/s_i] \qquad (2)$$

Let $p = 2q/m$. When t is zero this degenerates to the many-at-a-time relaxation algorithm. In the many-at-a-time modified relaxation algorithm, a positive constant c is subtracted from e_i to guarantee finite convergence in the linearly separable case. We do the analogous thing to (2) to obtain the many-at-a-time modified extended relaxation algorithm. The positive constant c is added to e_i when $e_i \geq 0$.

$$W_{\ell+1} = W_\ell - p[t\sum_{e_i \geq 0} (e_i+c)Y_i/s_i + \sum_{e_i < 0} (e_i-c)\, Y_i/s_i]$$

This algorithm can be changed in the usual way to the following one-at-a-time algorithm.

$$W_{k+1} = \begin{cases} W_k - tp(e_i + c)Y_i/s_i & \text{if } e_i \geq 0 \\ W_k - p(e_i - c)Y_i/s_i & \text{otherwise} \end{cases} \quad (3)$$

This is our modified extended relaxation algorithm. It is the major component of our extended centered accelerated relaxation algorithm, as explained at the beginning of this section.

F. Centered Overcentered Accelerated Relaxation. This algorithm carries out algorithm D (centered accelerated relaxation), overcenters the solution (that is, expands the dead zone a little more than centering would), carries out algorithm D again, overcenters again, etc. We now define overcentering. Let $f' = \min_{c_1}(e_i)$ Let g'' be the next strictly larger such minimum. Let $g' = \min_{c_2}(e_i)$. Let g'' be the next strictly larger such minimum. Suppose that $f'' - f' \geq g'' - g'$. (The other case is defined analogously.) Overcentering is defined as f' g'' - centering.

The centered overcentered accelerated relaxation algorithm uses the accelerated relaxation algorithm a total of kmax times, where kmax is typically five. The algorithm is the following:

(1) Set kmax.

(2) Set $W_0'' = (0,0,\ldots,0)$.

(3) Set $k = 1$.

(4) Starting with W_{k-1}'', let the solution found by the accelerated relaxation algorithm be W_k.

(5) Obtain W_k' by centering W_k.

(6) Type the centrality criteria c_1 and c_2 for W_k'. (See Section 6.)

(7) If $k = kmax$, type the best results (the results for the best C_1, and the results for the best C_2) and stop.

(8) Obtain W'' by overcentering W_k'.

(9) Set $k = k + 1$.

(10) Go to (4).

6. CENTRALITY CRITERIA

The concept of the central hyperplane has never been precisely defined. It remains an intuitive notion. Nevertheless, we need a definite measure of centrality in order to compare the various algorithms. We compromise by giving two distinct quantitative measures of centrality but we do not specify the relative importance of these criteria.

Suppose that we have a solution hyperplane $W \cdot Y - b = 0$. We assume that the solution has been broadened or centered. Our first centrality criterion is half the dead zone width $C_1(W) = b/v$. The larger this criterion is, the more central the solution hyperplane H tends to be.

We now obtain our second centrality criterion from Equation (1) for the extended relaxation risk. Recall that we apply a criterion only if the solution with dead zone correctly classifies every vector in the training set, that is, only if $e_i \geq 0$ for all i. We obtain $C_2(W) = t \sum_I e_i^2/s_i$. The smaller this criterion is, the more central the solution hyperplane tends to be.

7. EXPERIMENTS WITH THE ALGORITHMS.

Each of the six algorithms was run on each of three data sets. The iris data consists of four measurements on each of sixty irises, the first twenty from each of three classes. This is a subset of the famous set treated by R. A. Fisher [3]. The Fig. 9 data consists of the artificial set shown in Fig. 9. This set was chosen because it is simple to consider and yet it is difficult for many algorithms to get an optimal solution according to our criteria. The JU data consists of eight measurements on each of thirty hand printed J's and thirty hand printed U's. This set is described and used in [6].

The results for all three data sets are summarized in Table 2. Percentages are averaged over the three data sets. That is, each entry is a third of the sum of the three percent entries taken from Table 1 and the two tables for the iris and JU data sets.

8. CONCLUSIONS

The conclusions are based on experiments with three specific data sets and therefore may not be true in general. However, they do not go against intuition.

(1) The last three algorithms are substantially better than the first three. If computer time is available, one should try algorithms D, E, and F and take the best result obtained.

(2) If computer time is short, one should use algorithm D, which is fast and is central but not the most central solution.

(3) Although slower than algorithm D, algorithms E and F find the most central solutions.

(4) Centering improves the solution tremendously and takes very little time. This can be proved by comparing algorithm A and B and by comparing algorithms C and D in Table 2.

(5) Although frequently used, the modified relaxation algorithm is particularly bad. It is slow and finds solutions that are not at all central.

(6) The broadened accelerated relaxation algorithm is the fastest, but the solutions are not at all central.

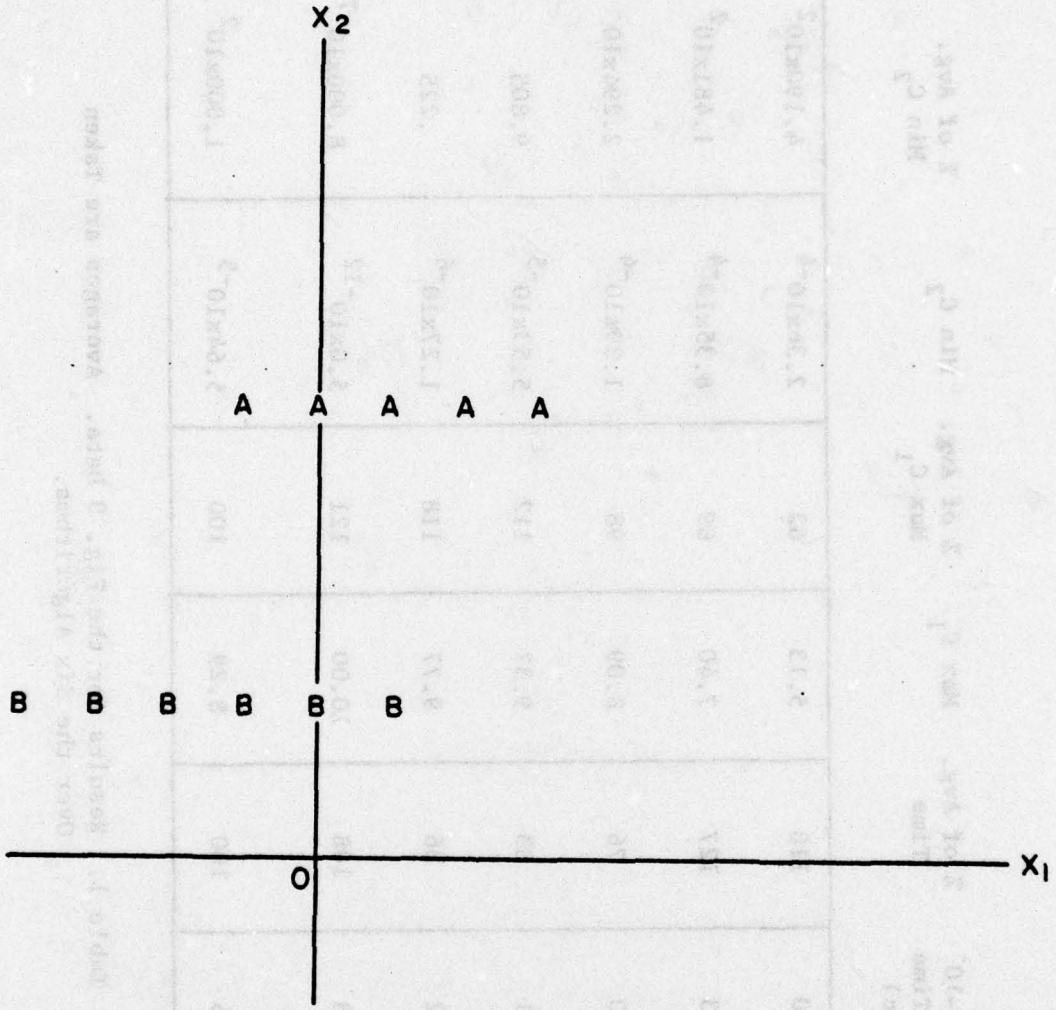

Fig. 9 - Artificial data set. The members of Class A are (-5, 30), (0, 30), (5, 30), (10, 30), and (15, 30). The members of Class B are (-20, 10), (-15, 10), (-10, 10), (-5, 10), (0, 10), and (5, 10).

Kind of Relaxation Algorithm	DEC-10 CPU Time (sec)	% of Avg. Time	Max C_1	% of Avg. Max C_1	Min C_2	% of Avg. Min C_2	Dist. fm. Origin
Broadened Modified	3.0	118	5.13	62	2.36×10^{-4}	4.190×10^{2}	16.28
Centered Modified	3.3	127	7.40	89	8.35×10^{-4}	1.481×10^{2}	18.54
Broadened Accelerated	2.0	76	8.09	98	1.29×10^{-4}	2.294×10	18.62
Centered Accelerated	2.1	83	9.32	112	5.53×10^{-5}	9.805	19.85
Extended Centered Accelerated	2.2	86	9.77	118	1.27×10^{-6}	.225	20.03
Centered Over Centered Accelerated	2.8	108	10.00	121	5.0×10^{-12}	8.000×10^{-7}	20.00
Average	2.6	100	8.29	100	5.64×10^{-5}	1.000×10^{2}	18.89

Table 1. Results for the Fig. 9 Data. Averages are Taken Over the Six Algorithms.

Kind of Relaxation Algorithm	Avg. % for time	Avg. % for C_1	Avg. % for C_2
Broadened Modified	110	75	212.1
Centered Modified	118	104	83.9
Broadened Accelerated	63	54	272.9
Centered Accelerated	71	118	16.0
Extended Centered Accelerated	114	123	6.3
Centered Over Centered Accelerated	125	126	8.8

Table 2. Summary of Results. Percentages are Averaged Over the Three Data Sets.

REFERENCES

1. Chang, C. L., The Accelerated Relaxation Method for Linear Inequalities. IEEE Trans. on Computers, Vol. C-20, No. 2, Feb. 1971, pp. 220-225.

2. Duda, R. and Hart, P., Pattern Classification and Scene Analysis. Wiley Interscience, New York, 1973.

3. Fisher, R. A., The Use of Multiple Measurements in Taxonomic Problems. Ann. Eugen., Vol. 7, 1936, pp. 178-188. Reprinted in Contributions to Mathematical Statistics, Wiley, New York, 1950, pp. 32.179 - 32.188.

4. Mays, C. H., Effects of Adaptation Parameters on Convergence Time and Tolerance for Adaptive Threshold Elements, IEEE Trans. on Elec. Computers, Aug. 1964, pp. 465-468.

5. Meisel, W., Computer Oriented Approaches to Pattern Recognition, Academic Press, New York, 1972.

6. Slagle, J., and Lee, R.C.T., Application of Game Tree Searching Techniques to Sequential Pattern Recognition, CACM, Vol. 14, no. 2, Feb. 1971, pp. 103-118.

7. Slagle, J., Finding Central Hyperplanes for Automatic Pattern Recognition, Heuristics Laboratory Internal Report #17, National Institutes of Health, Bethesda, Maryland, 1974.

8. Yue, S. F., A Method to Obtain Central Hyperplanes, MS Thesis in Applied Mathematics, National Tsing Hua University, Taiwan, June 1976.